

Vectorul de frecvențe și aplicații

Vectorul de frecvențe reține numărul de apariții al fiecărei valori citite într-un vector. Folosirea vectorului de frecvențe permite scrierea unor algoritmi eficienți în cazul în care datele de intrare au valori dintr-un domeniu cunoscut care poate fi prelucrat rapid. Folosirea unui vector de frecvență sau marcarea este eficientă numai în cazul în care valorile care interesează sunt întregi și numărul valorilor distincte posibile este cel mult 1.000.000, pentru un timp maxim de 1 sec/test.

De exemplu acest vector poate fi folosit pentru sortarea rapidă a datelor, în timp liniar.

Tot cu ajutorul acestui vector pot fi implementate **operațiile de bază cu mulțimi**:

- **Căutarea** unui element
- **Intersecția** a două (sau mai multe) mulțimi
- **Reuniunea** a două (sau mai multe) mulțimi

În orice mulțime elementele sunt unice, iar vectorul frecvențelor are doar valori 0 sau 1. Acest vector este numit **vectorul caracteristic** al unei mulțimi. Vectorul de frecvențe poate fi folosit pentru a obține rapid mulțimea asociată ca un vector caracteristic astfel:

- 0 înseamnă că elementul nu aparține mulțimii
- o valoare diferită de 0 înseamnă că elementul aparține mulțimii și indică numărul de apariții

APLICAȚII ale vectorului de frecvențe sau vectorului caracteristic al unei mulțimi:

1. Prelucrarea cifrelor
2. Sortare în timp liniar
3. Operații cu mulțimi

1. Prelucrarea cifrelor unui număr

Aplicație rezolvată [Cifre distincte](#):

Fiind dat un număr natural n între 1 și două miliarde să se afișeze cifrele și numărul de elemente al mulțimii cifrelor sale.

Date de intrare

Fișierul de intrare `cfdist.in` conține numărul n .

Date de ieșire

Fișierul de ieșire `cfdist.out` va conține un singur număr, numărul de cifre distincte ale lui n .

Restricții

- $1 \leq n \leq 2.000.000.000$

Exemplu

<code>cfdist.in</code>	<code>cfdist.out</code>
234234234	3

Explicație

Numărul 234234234 conține 3 cifre distincte, 2, 3 și 4.

Vectorul de frecvență pentru cifrele unui număr se declară sub forma unui vector cu 10 componente, de la $a[0], \dots, a[9]$. Acestea vor fi inițializate cu 0, iar după citirea numărului n se va incrementa cu 1 numărul aparițiilor pentru fiecare cifră a lui n .

Atenție! Vectorul de frecvențe nu permite refacerea numărului citit inițial, dacă este necesară valoarea acestuia trebuie memorată separat.

Soluție: Codul C++ este dat în continuare:

```
1. #include<iostream>
2. using namespace std;
3. int a[10],n;
4.
5. int main()
6. {
7.     int i,nn,nc=0;
8.     cout<<"n=";
9.     cin>>n;
10.    nn=n;//clonam valoarea n in variabila nn
11.    while(nn>0) //construim vectorul frecventelor
12.    { a[nn%10]++;
13.      nn=nn/10;}
14.    //afisam cifrele distincte
15.    for(i=0;i<10;i++)
16.        if (a[i]>0) {cout<<i;nc++;}
17.    cout<<"\n"<<nc<<" cifre distincte";
18.    return 0;
19. }
```

Probleme propuse:

1. **Cifre comune:** Se citesc două numere naturale n și m . Să se afișeze numărul de cifre distincte comune ambelor numere.

Indicație de rezolvare: Construim vectorii de frecvență pentru fiecare număr și apoi aflăm cifrele comune. Dacă notăm cu a vectorul de frecvențe al lui n și cu b vectorul de frecvențe al lui m condiția ca cifra c să fie comună este $a[c]>0 \ \&\& \ b[c]>0$.

2. **Maxnr:** Dat un număr natural n între 1 și două miliarde să se afișeze cel mai mare număr care se poate forma cu cifrele sale.

Indicație de rezolvare: Construim vectorul de frecvențe asociat numărului n , notat cu a . Numărul maxim care poate fi format cu cifrele sale se obține prin parcurgerea în ordine descrescătoare, de la 9 la 0, a vectorului de frecvențe și scriind fiecare cifră c de $a[c]$ ori.

3. **Minnr:** Dat un număr natural n între 1 și două miliarde să se afișeze cel mai mic număr care se poate forma cu cifrele sale.

Indicație de rezolvare: Construim vectorul de frecvențe asociat numărului n , notat cu a . Avem două situații posibile: a) numărul n nu are cifra 0 sau b) numărul n conține și cifra 0.

- a) Numărul minim se obține prin parcurgerea în ordine crescătoare, de la 1 la 9, a vectorului frecvențelor și afișarea fiecărei cifre c de $a[c]$ ori.
- b) Se determină prima cifră nenulă a lui n și se afișează ca prima cifră a rezultatului. După prima cifră se afișează toate zerourile care apar în n , iar apoi se aplică același algoritm ca în cazul a.

4. [Minnrk](#): Se citesc două numere, n și k . Să se afișeze cel mai mic număr de k cifre care se poate forma cu cifrele lui n .

Indicație de rezolvare: Construim vectorul de frecvențe asociat numărului n , notat cu a . Avem două situații posibile: a) numărul n nu are cifra 0 sau b) numărul n conține și cifra 0. Aplicăm același algoritm de la problema Minnr, dar afișăm doar primele k cifre din numărul rezultat.

5. [Cifre1](#): Se dau două numere naturale a și b cu maxim 9 cifre.

- a) Să se determine cifrele distincte, comune numerelor a și b .
b) Să se afișeze numărul cel mai mare format din toate cifrele lui a și b .

2. Sortare în timp liniar

Culori (clasa a 5-a)

Se dă o secvență de n culori codificate cu numere între 1 și 99. Să se afișeze culorile în ordinea lor crescătoare. Acesta este un exercițiu introductiv în folosirea vectorilor de frecvență.

Date de intrare

Fișierul de intrare `culori.in` va conține pe prima linie numărul n . Pe a doua linie va conține n numere despărțite prin spațiu.

Date de ieșire

În fișierul de ieșire `culori.out` veți scrie o singură linie conținând cele n culori în ordine crescătoare.

Restricții

- $1 \leq n \leq 1.000.000$
- $1 \leq \text{culoare} \leq 99$

Exemplu

culori.in	culori.out
10 4 7 3 0 3 7 4 0 2 2	0 0 2 2 3 3 4 4 7 7

Explicație

Cele zece culori de la intrare au fost afișate în ordine crescătoare.

Soluție

Construim vectorul de frecvență a cifrelor $a[0], \dots, a[9]$ și afișăm cifrele în ordine crescătoare de la 0 la 9, iar fiecare cifră c va fi scrisă de $a[c]$ ori.

3. Operații cu mulțimi

- A. Într-o mulțime în sens matematic - **set**, elementele au valori distincte iar vectorul de frecvențe se numește **vector caracteristic**. Dacă mulțimea A are valorile din mulțimea $\{0,1,2,\dots,n\}$ vectorul caracteristic al mulțimii A este definit prin:

$$a[i] = \begin{cases} 1, & \text{daca } i \in A \\ 0, & \text{daca } i \notin A \end{cases}$$

1. Inițializarea vectorului caracteristic:

```
1. void citire(int &n, int a[])
2. {
3.     int i,x;
4.     for(i=1;i<=n;i++)
5.     {cin>>x;
6.       a[x]++;}
7. }
```

2. Testarea apartenenței unui element x la mulțimea A:

```
1. int apartine(int x,int a[])
2. {
3.     if(a[x]>0) return 1;
4.     else return 0;
5. }
```

3. Intersecția a două mulțimi A și B de numere naturale din intervalul închis $[0,n]$.

$$A \cap B = \{x \in A \text{ și } x \in B\}$$

```
1. void intersecție(int n,int a[],int b[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]*b[i]>0) cout<<i;
5. }
```

4. Reuniunea a două mulțimi A și B de numere naturale din intervalul închis $[0,n]$.

$$A \cup B = \{x \in A \text{ sau } x \in B\}$$

```
1. void reuniune(int n,int a[],int b[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]+b[i]>0) cout<<i;
5. }
```

5. Diferența a două mulțimi A și B de numere naturale din intervalul închis $[0,n]$.

$$A \setminus B = \{x \in A \text{ și } x \notin B\}$$

```
1. void diferența(int n,int a[],int b[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]>0 && b[i]==0) cout<<i;
5. }
```

- B. O *mulțime generalizată* sau **multiset** este o colecție de elemente în care este permis ca o valoare să se repete. Valorile care se repetă pot fi considerate *identice* sau *echivalente*. De exemplu, într-o listă de persoane pot să existe mai multe persoane care au aceeași înălțime sau cu același prenume.

Vectorul de frecvență reține *numărul de apariții* al unei valori din multiset:

$a[i]$ = numărul de apariții al valorii i în multiset

Citirea și testul de apartenență sunt identice cu cele de la mulțimi.

1. Intersecția a două mulțimi generalizate A și B de numere naturale din intervalul închis $[0,n]$.

```
1. void intersecție(int n,int a[],int b[],int c[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]<=b[i]) c[i]=a[i];
5.     else c[i]=b[i];
6. }
```

2. Reuniunea a două mulțimi generalizate A și B de numere naturale din intervalul închis $[0,n]$.

```
1. void reuniune(int n,int a[],int b[],int c[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]<=b[i]) c[i]=b[i];
5.     else c[i]=a[i];
6. }
```

3. Diferența a două mulțimi generalizate A și B de numere naturale din intervalul închis $[0,n]$.

```
1. void diferența(int n,int a[],int b[],int c[])
2. { int i;
3.   for(i=0;i<=n;i++)
4.     if(a[i]>b[i]) c[i]=a[i]-b[i];
5.     else c[i]=0;
6. }
```

4. Sortarea crescătoare elementelor unei mulțimi generalizate se face printr-o simplă parcurgere a intervalului $[0,n]$.

```
1. void tipar(int n,int a[])
2. { int i,j;
3.   for(i=0;i<=n;i++)
4.     for(j=0;j<=a[i];j++)
5.       cout<<i<<" ";
6. }
```